**微服务架构下的分布式事务解决方案**

1. **微服务的发展**

微服务倡导将复杂的单体应用拆分为若干个功能简单、松耦合的服务，这样可以降低开发难度、增强扩展性、便于敏捷开发。当前被越来越多的开发者推崇，很多互联网行业巨头、开源社区等都开始了微服务的讨论和实践。Hailo有160个不同服务构成，NetFlix有大约600个服务。国内方面，阿里巴巴、腾讯、360、京东、58同城等很多互联网公司都进行了微服务化实践。当前微服务的开发框架也非常多，比较著名的有Dubbo、SpringCloud、thrift 、grpc等。

1. **微服务落地存在的问题**

虽然微服务现在如火如荼，但对其实践其实仍处于探索阶段。很多中小型互联网公司，鉴于经验、技术实力等问题，微服务落地比较困难。如著名架构师Chris Richardson所言，目前存在的主要困难有如下几方面：

1）单体应用拆分为分布式系统后，进程间的通讯机制和故障处理措施变的更加复杂。

2）系统微服务化后，一个看似简单的功能，内部可能需要调用多个服务并操作多个数据库实现，服务调用的分布式事务问题变的非常突出。

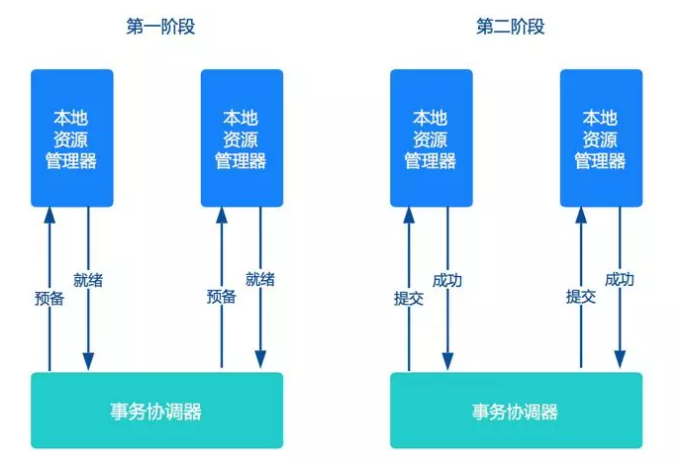
3）微服务数量众多，其测试、部署、监控等都变的更加困难。

随着RPC框架的成熟，第一个问题已经逐渐得到解决。例如dubbo可以支持多种通讯协议，springcloud可以非常好的支持restful调用。对于第三个问题，随着docker、devops技术的发展以及各公有云paas平台自动化运维工具的推出，微服务的测试、部署与运维会变得越来越容易。

而对于第二个问题，现在还没有通用方案很好的解决微服务产生的事务问题。分布式事务已经成为微服务落地最大的阻碍，也是最具挑战性的一个技术难题。 为此，本文将深入和大家探讨微服务架构下，分布式事务的各种解决方案，并重点为大家解读阿里巴巴提出的分布式事务解决方案----GTS。该方案中提到的GTS是全新一代解决微服务问题的分布式事务互联网中间件。

1. **传统分布式事务解决方案**
   1. **基于XA协议的两阶段提交方案**

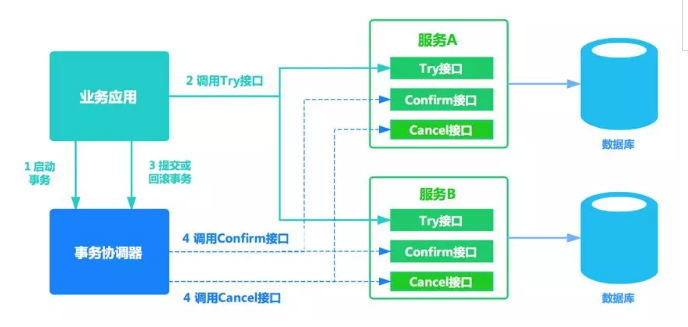
交易中间件与数据库通过 XA 接口规范，使用两阶段提交来完成一个全局事务， XA 规范的基础是两阶段提交协议。 第一阶段是表决阶段，所有参与者都将本事务能否成功的信息反馈发给协调者；第二阶段是执行阶段，协调者根据所有参与者的反馈，通知所有参与者，步调一致地在所有分支上提交或者回滚。



两阶段提交方案应用非常广泛，几乎所有商业OLTP数据库都支持XA协议。但是两阶段提交方案锁定资源时间长，对性能影响很大，基本不适合解决微服务事务问题。

* 1. **TCC方案**

TCC方案在电商、金融领域落地较多。TCC方案其实是两阶段提交的一种改进。其将整个业务逻辑的每个分支显式的分成了Try、Confirm、Cancel三个操作。Try部分完成业务的准备工作，confirm部分完成业务的提交，cancel部分完成事务的回滚。基本原理如下图所示。



事务开始时，业务应用会向事务协调器注册启动事务。之后业务应用会调用所有服务的try接口，完成一阶段准备。之后事务协调器会根据try接口返回情况，决定调用confirm接口或者cancel接口。如果接口调用失败，会进行重试。

TCC方案让应用自己定义数据库操作的粒度，使得降低锁冲突、提高吞吐量成为可能。 当然TCC方案也有不足之处，集中表现在以下两个方面：

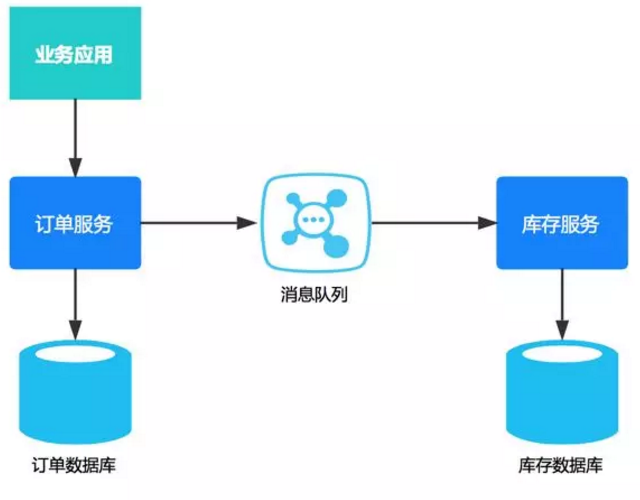
对应用的侵入性强。业务逻辑的每个分支都需要实现try、confirm、cancel三个操作，应用侵入性较强，改造成本高。

实现难度较大。需要按照网络状态、系统故障等不同的失败原因实现不同的回滚策略。为了满足一致性的要求，confirm和cancel接口必须实现幂等。

上述原因导致TCC方案大多被研发实力较强、有迫切需求的大公司所采用。微服务倡导服务的轻量化、易部署，而TCC方案中很多事务的处理逻辑需要应用自己编码实现，复杂且开发量大。

* 1. **基于消息的最终一致性方案**

消息一致性方案是通过消息中间件保证上、下游应用数据操作的一致性。基本思路是将本地操作和发送消息放在一个事务中，保证本地操作和消息发送要么两者都成功或者都失败。下游应用向消息系统订阅该消息，收到消息后执行相应操作。



消息方案从本质上讲是将分布式事务转换为两个本地事务，然后依靠下游业务的重试机制达到最终一致性。基于消息的最终一致性方案对应用侵入性也很高，应用需要进行大量业务改造，成本较高。

1. **GTS－阿里分布式事务解决方案**

参考GTS阿里分布式事务解决方案